4. Paulina

Program Name: Paulina.java **Input File: paulina.dat**

In the last weekend campout with her Scouting Venture Crew, Paulina learned about orienteering, but still needs a little bit of help. In the training, instead of the traditional value of zero being north, zero was considered to be east. The direction 360 was also east, as was -360. Due west was the positive or negative 180. Due north was either 90 or -270. To make it easier, the orienteering course only dealt with directions that were multiples of 30 or 45, which made it easy for her to use the special 45-45-90 and 30-60-90 triangle formulas she had just learned in algebra.

The purpose of the course was to predict an ending position on the coordinate grid, with (0,0) being the home base location, and all other positions relative to home base. The instructions given were in numeric pairs, each representing a vector, the first value indicating the distance traveled in miles, and the second the direction traveled. The task was to predict the ending location of the course.

Several vectors could be given, representing multiple movements in one trek. For example, the vector (5, 45) meant to go 5 miles in a north-east direction, which would end up at position (3.5355, 3.5355) relative to home. An additional vector of (4, -30) in the same course would start from (3.5355, 3.5355), go 4 miles in the direction 30 degrees below east, ending up at (6.9996, 1.5355).

Input - Several sets of data, each representing one trek, and each on one line with single space separation. Each data set consists of an initial value N, followed by N vector integer pairs, each pair consisting of a distance in miles, and a positive or negative direction (-360 <= direction <= 360) relative to East, as described above. Each direction is guaranteed to be a positive or negative multiple of 30 or of 45.

Output - An ordered pair representing the final **(x, y)** position upon completion of the trek as designated by the data set, enclosed within parentheses, with a single space after the comma, each coordinate value output rounded to a precision of 4 decimal places. A tolerance of +-0.0001 will be accepted for either value.

Sample data:

```
2 5 45 4 -30
4 2 90 2 180 2 270 2 0
3 2 120 2 -330 2 60
```

Sample Output:

```
(6.9996, 1.5355)
(0.0000, 0.0000)
(1.7321, 4.4641)
```